US006073206A

# United States Patent [19]

**Piwonka et al.**

[11] **Patent Number: 6,073,206**

[45] **Date of Patent: Jun. 6, 2000**

[54] **METHOD FOR FLASHING ESCD AND VARIABLES INTO A ROM**

[75] Inventors: **Mark A. Piwonka**; **Louis B. Hobson**, both of Tomball; **Jeffrey D. Kane**, Spring; **Randall L. Hess**, Cypress, all of Tex.

[73] Assignee: **Compaq Computer Corporation**, Houston, Tex.

[21] Appl. No.: **09/070,866**

[22] Filed: **Apr. 30, 1998**

[51] **Int. Cl.**[7] ..... **G06F 13/00**

[52] **U.S. Cl.** ..... **711/102**; 711/103; 711/162; 711/170; 714/6; 395/653; 707/202; 707/203; 707/204

[58] **Field of Search** ..... 711/102, 103, 711/161, 162, 170; 714/4, 5, 6; 395/653; 707/202, 203, 204

[56] **References Cited**

U.S. PATENT DOCUMENTS

| | | | |
|---|---|---|---|
| 5,339,437 | 8/1994 | Yuen | 395/700 |
| 5,608,876 | 3/1997 | Cohen et al. | 710/101 |
| 5,724,027 | 3/1998 | Shipman et al. | 340/825.31 |
| 5,748,940 | 5/1998 | Angelo et al. | 711/163 |
| 5,864,698 | 1/1999 | Krau et al. | 713/2 |

OTHER PUBLICATIONS

Compaq Computer Corporation, Phoenix Technologies Ltd., Intel Corporation, "'Preliminary' Plug and Play Bios Specification, Version 1.0A, " Mar. 10, 1994, pp. 7–23, 57–64.

Compaq Computer Corporation, Intel Corporation, Phoenix Technologies, Ltd., Extended System Configuration Data Specification, Version 1.02, Feb. 14, 1994, ©1993, pp. 1–33.

Advanced Micro Devices, Inc., "Am291F002T/Am29F002B, 2 Megabit (262,144×8–Bit) CMOS 5.0 Volt–only, Sector Architecture Flash Memory," © 1996, pp. 1–34.

Intel Corporation, "Pentium™ Processor's User's Manual, vol. 3: Architecture and Programming Manual," © Intel Corporation, 1996, pp. 20.1–20.9.

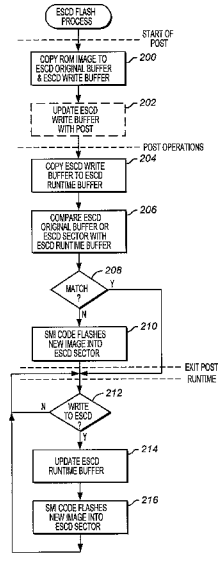
Intel Corporation, "Intel486™ SL Microprocessor SuperSet Programmer's Reference Manual," Nov., 1992, pp. 6.28–6.53.

*Primary Examiner*—Do Hyun Yoo
*Assistant Examiner*—Nasser Moazzami
*Attorney, Agent, or Firm*—Akin, Gump, Strauss, Hauer & Feld, LLP

[57] **ABSTRACT**

A computer system for flashing Extended System Configuration Data (ESCD) and associated variables to a flash read-only memory (ROM) is provided. During Power-On-Self-Test (POST) code, a ROM image is copied from an ESCD sector of a read-only memory to an ESCD original buffer and an ESCD write buffer. The ESCD write buffer may be updated by POST code. Following the POST operations, the contents of the ESCD write buffer are copied to an ESCD runtime buffer. The contents of the ESCD original buffer or the ESCD sector are compared to the contents of the ESCD runtime buffer. If the contents of the ESCD runtime buffer differ from the contents of the compared buffer or sector, SMI code flashes the ROM image in the ESCD runtime buffer to the flash ROM. If the ESCD runtime buffer is the same as the contents of the compared buffer or sector, a ROM flash it not performed. POST is then exited and the computer system is booted. During runtime, if a write is performed to ESCD data or an associate variable, the ESCD runtime buffer is updated with the ESCD data or variable provided for the write operation. The SMI code then flashes the ROM image in the ESCD runtime buffer into the ESCD sector to save the ESCD data or variable provided by the write operation.

**24 Claims, 3 Drawing Sheets**



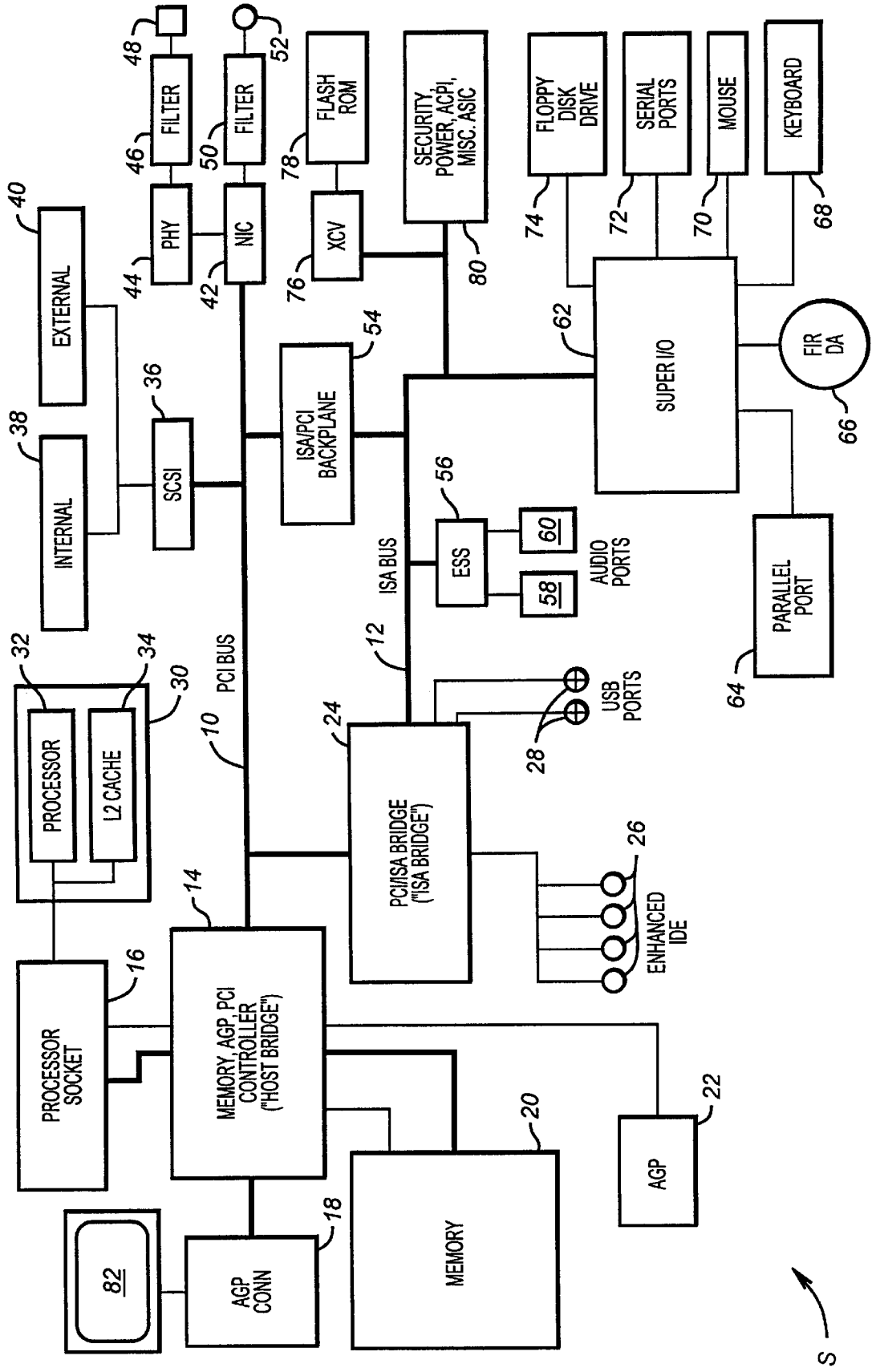

PROCESSOR
L2 CACHE
PROCESSOR SOCKET
MEMORY, AGP, PCI CONTROLLER ("HOST BRIDGE")
PCI/ISA BRIDGE ("ISA BRIDGE")
PCI BUS
ISA BUS
ISA/PCI BACKPLANE
SCSI
EXTERNAL
INTERNAL
PHY
NIC
FILTER
FILTER
XCV
FLASH ROM
SECURITY, POWER, ACPI, MISC. ASIC
FLOPPY DISK DRIVE
SERIAL PORTS
MOUSE
KEYBOARD
SUPER I/O
FIR
DA
ESS
AUDIO PORTS
USB PORTS
ENHANCED IDE
PARALLEL PORT
AGP
MEMORY
AGP CONN
82
S

FIG. 1

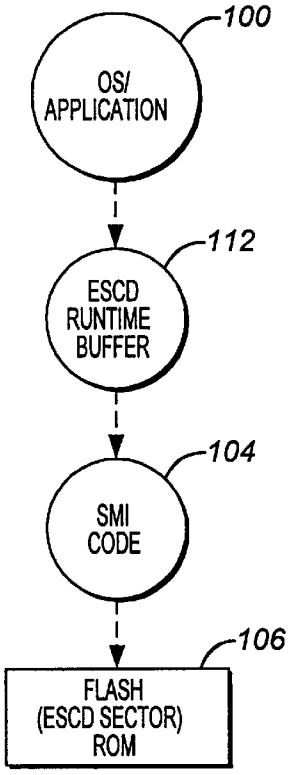

100
OS/
APPLICATION
112
ESCD
RUNTIME
BUFFER
104
SMI
CODE
106
FLASH
(ESCD SECTOR)
ROM

FIG. 2

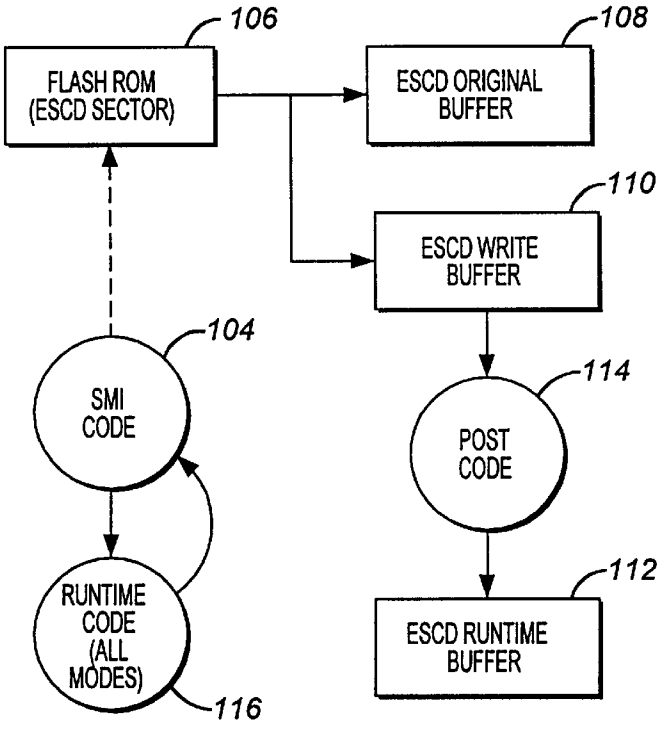

106
FLASH ROM
(ESCD SECTOR)
108
ESCD ORIGINAL
BUFFER
110
ESCD WRITE
BUFFER
104
SMI
CODE
114
POST
CODE
112
ESCD RUNTIME
BUFFER
RUNTIME
CODE
(ALL
MODES)
116

FIG. 3

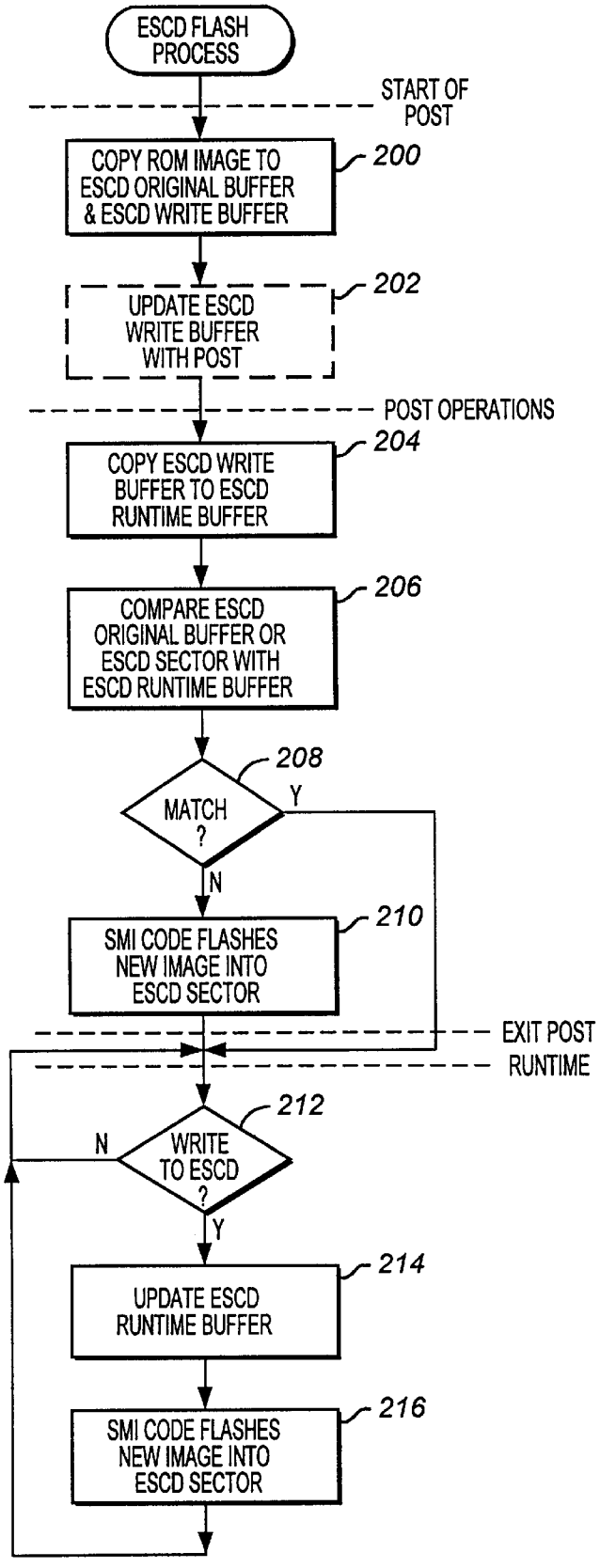

ESCD FLASH PROCESS
START OF POST
COPY ROM IMAGE TO ESCD ORIGINAL BUFFER & ESCD WRITE BUFFER
200
UPDATE ESCD WRITE BUFFER WITH POST
202
POST OPERATIONS
COPY ESCD WRITE BUFFER TO ESCD RUNTIME BUFFER
204
COMPARE ESCD ORIGINAL BUFFER OR ESCD SECTOR WITH ESCD RUNTIME BUFFER
206
MATCH ?
208
Y
N
SMI CODE FLASHES NEW IMAGE INTO ESCD SECTOR
210
EXIT POST
RUNTIME
WRITE TO ESCD ?
212
N
Y
UPDATE ESCD RUNTIME BUFFER
214
SMI CODE FLASHES NEW IMAGE INTO ESCD SECTOR
216

FIG. 4

# METHOD FOR FLASHING ESCD AND VARIABLES INTO A ROM

## BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to read-only-memory (ROM) flashing, and more particularly to a method for flashing Extended System Configuration Data (ESCD) and variables into a ROM of a computer system.

2. Description of the Related Art

In order to support the automatic configuration of Plug and Play devices on platforms that include a standard expansion bus, non-volatile storage such as CMOS RAM, ROM, or NVRAM is used to store information about system resources used by non-Plug and Play devices in a computer system. This configuration information is typically stored in an Extended System Configuration Data (ESCD) format. The ESCD format accommodates storage of configuration information for Plug and Play devices and non-Plug and Play devices. ESCD generally allows Plug and Play System Basic Input Output Services (BIOS) code to more fully configure a computer system at power up by specifying the system resources assigned to devices that have been installed in the system. The portion of non-volatile storage used to store the ESCD may be only a subset of the total non-volatile storage available in a computer system.

System BIOS code is used by the operating system to read or write ESCD as the non-volatile storage storing ESCD has traditionally resided in the system BIOS memory space. Accordingly, only system BIOS has known the location of the ESCD in the non-volatile storage and the proper method for accessing the non-volatile storage. If the non-volatile storage is memory-mapped, the physical base address of the memory mapped non-volatile storage allows a "caller," such as an operating system, to construct a segment descriptor pointed to by a memory pointer known as a segment selector. If a function to read or write ESCD is called from a protected mode, the segment descriptor is created from a protected mode base segment address typically termed an ESCD Selector parameter. If a function to read or write ESCD is called from real mode, a segment descriptor is created from a real mode base segment address typically termed a BIOS Selector parameter. A segment selector serves as a pointer to a segment descriptor for an ESCD area of non-volatile storage. In addition, a function to read or write ESCD generates a pointer to the size of ESCD for determining an entry point into the ESCD. A function to read ESCD transfers ESCD from an ESCD area of non-volatile storage to a memory buffer typically having a pointer termed ESCD Buffer, and a function to write ESCD transfers data from the memory buffer to an ESCD area of non-volatile storage. ESCD functions are further described in the Extended System Configuration Data Specification, Version 1.02, published on Feb. 14, 1994, by Compaq Computer Corporation, Intel Corporation, and Phoenix Technologies, Ltd.

ESCD may be updated during a Power-On-Self-Test (POST) or at runtime. For example, ESCD may be updated by POST if POST detects that a new Plug and Play bootable device is added to the computer system. ESCD information may be used by POST to allocate system resources to all configurable devices that are known to the system BIOS. System resources typically include Direct Memory Access (DMA) channels, Interrupt Request Lines (IRQ's), Input/Output (I/O) addresses, and memory. ESCD also may be updated during runtime by system software in order to effect configuration of devices on the next boot. An ESCD interface provides a mechanism for allowing system software to lock system resources allocated to specific devices in the system and thereby avoid system resource conflicts.



Updating ESCD in a conventional computer system has been dependent upon a segment selector from an operating system. The segment selector used by system BIOS to address an ESCD area of non-volatile storage is generated by the operating system. As a segment selector is exclusively configured for the memory map of the operating system, a segment selector loses meaning outside the context of the operating system memory map. A segment selector provided by an operating system also must be configured for the particular operating mode—real mode, 16-bit protected mode, or 32-bit protected mode—whichever is applicable. The segment selector in one operating mode is therefore different from the segment selector for another operating mode. Further, the need for a segment selector from an operating system to update ESCD has required a different selector if an ESCD area is relocated. Another drawback of using a segment selector from an operating system to read or write ESCD is the need to use a relatively slow bus within a computer system to reach the ESCD area of non-volatile storage.

ESCD has typically resided in a portion of a ROM treated as an option ROM. Option ROM represents a designated area of ROM space which an operating system preferably is programmed not to utilize. In certain operating modes, however, availability of an option ROM is not guaranteed. For example, during runtime in a 32-bit protected mode, an operating system may not provide a segment selector to an option ROM, thereby preventing access to the option ROM during runtime. In addition, the size of the ROM space for which an operating system provides a segment selector is limited.

## SUMMARY OF THE INVENTION

Briefly, a computer system according to the present invention provides a method of flashing Extended System Configuration Data (ESCD) and variables stored with ESCD to a read-only-memory (ROM) using System Management Interrupt (SMI) code. An ESCD sector is defined in a read-only-memory for storing ESCD and associated variables. During Power-On-Self-Test (POST) code, a ROM image of the ESCD sector is copied to an ESCD original buffer and an ESCD write buffer. The ESCD original buffer and ESCD write buffer are areas of random access memory (RAM) used during POST for storing a ROM image of ESCD and associated variables. The ESCD write buffer may be updated by POST code. Following the POST operations, the contents of the ESCD write buffer are copied to an ESCD runtime buffer. The ESCD runtime buffer is a RAM area to be used during runtime for storing ESCD data and associated variables. The contents of the ESCD original buffer or the ESCD sector are compared to the contents of the ESCD runtime buffer. If the contents of the ESCD runtime buffer differ from the contents of the compared buffer or sector, SMI code flashes the ROM image in the ESCD runtime buffer to the ESCD sector. If the ESCD runtime buffer is the same as the contents of the compared buffer or sector, a ROM flash is not performed. POST is then exited and the computer system is booted.

During runtime, any write to ESCD or an associated variable is detected. If a write to ESCD or an associated variable is detected, the ESCD runtime buffer is updated with the ESCD or variable provided for the write operation.

The SMI code then flashes the ROM image in the ESCD runtime buffer into the ESCD sector to save the ESCD or variable provided by the write operation.

This method of flashing ESCD and associated variables permits ESCD and associated variables to be flashed in any operating mode whether real mode, 16-bit protected mode, or 32-bit protected mode. An operating mode-specific segment selector from an operating system for locating ESCD thus is no longer needed. Another advantage of this method of flashing ESCD and associated variables is the reduced POST execution time. This advantage is achieved by using the ESCD buffers to read a ROM image during late POST and runtime rather than reading a ROM image from the ESCD sector.

## BRIEF DESCRIPTION OF THE DRAWINGS

A better understanding of the present invention can be obtained when the following detailed description of the preferred embodiment is considered in conjunction with the following drawings, in which:

FIG. **1** is a schematic diagram of a computer system including a flash ROM;

FIG. **2** is a data flow diagram of an ESCD flash process in accordance with the present invention;

FIG. **3** is a data flow diagram of an ESCD flash process using an ESCD original buffer, ESCD write buffer, and ESCD runtime buffer in accordance with the present invention; and

FIG. **4** is a flowchart of the ESCD flash process of FIG. **3**.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Turning to FIG. **1**, illustrated is a typical computer system S implemented according to the invention. While this system is illustrative of one embodiment, the techniques according to the invention can be implemented in a wide variety of systems. The computer system S in the illustrated embodiment is a PCI bus/ISA bus based machine, having a peripheral component interconnect (PCI) bus **10** and an industry standard architecture (ISA) bus **12**. The PCI bus **10** is controlled by PCI controller circuitry located within a memory/accelerated graphics port (AGP)/PCI controller **14**. This controller **14** (the "host bridge") couples the PCI bus **10** to a processor socket **16** via a host bus, an AGP connector **18**, a memory subsystem **20**, and an AGP **22**. A second bridge circuit, a PCI/ISA bridge **24** (the "ISA bridge") bridges between the PCI bus **10** and the ISA bus **12**.

The host bridge **14** in the disclosed embodiment is a 440LX Integrated Circuit by Intel Corporation, also known as the PCI AGP Controller (PAC). The ISA bridge **24** is a PIIX4, also by Intel Corporation. The host bridge **14** and ISA bridge **24** provide capabilities other than bridging between the processor socket **16** and the PCI bus **10**, and the PCI bus **10** and the ISA bus **12**. Specifically, the disclosed host bridge **14** includes interface circuitry for the AGP connector **18**, the memory subsystem **20**, and the AGP **22**. The ISA bridge **24** further includes an internal enhanced IDE controller for controlling up to four enhanced IDE drives **26**, and a universal serial bus (USB) controller for controlling USB ports **28**.

The host bridge **14** is preferably coupled to the processor socket **16**, which is preferably designed to receive a Pentium II processor module **30**, which in turn includes a microprocessor core **32** and a level two (L2) cache **34**. The processor socket **16** could be replaced with different processors other than the Pentium II without detracting from the spirit of the invention.

The host bridge **14**, when the Intel 440LX North Bridge is employed, supports extended data out (EDO) dynamic random access memory (DRAM) and synchronous DRAM (SDRAM), a 64/72-bit data path memory, a maximum memory capacity of one gigabyte, dual inline memory module (DIMM) presence detect, eight row address strobe (RAS) lines, error correcting code (ECC) with single and multiple bit error detection, read-around-write with host for PCI reads, and 3.3 volt DRAMs. The host bridge **14** support up to 66 megahertz DRAMs, whereas the processor socket **16** can support various integral and non-integral multiples of that speed.

The ISA bridge **24** also includes enhanced power management. It supports a PCI bus at 30 or 33 megahertz and an ISA bus **12** at ¼ of the PCI bus frequency. PCI revision 2.1 is supported with both positive and subtractive decode. The standard personal computer input/output (I/O) functions are supported, including a dynamic memory access (DMA) controller, two 82C59 interrupt controllers, an 8254 timer, a real time clock (RTC) with a 256 byte couple metal oxide semiconductor (CMOS) static RAM (SRAM), and chip selects for system read only memory (ROM), real time clock (RTC), keyboard controller, an external microcontroller, and two general purpose devices. The enhanced power management within the ISA bridge **24** includes fill clock control, device management, suspend and resume logic, advanced configuration and power interface (ACPI), and system management bus (SMBus) control, which implement the inter-integrated circuit (I2C) protocol.

The PCI bus **10** couples a variety of devices that generally take advantage of a high speed data path. This includes a small computer system interface (SCSI) controller **26**, with both an internal port **38** and an external port **40**. In the disclosed embodiment, the SCSI controller **26** is a AIC-7860 SCSI controller. Also coupled to the PCI bus **10** is a network interface controller (NIC) **42**, which preferably supports the ThunderLanTN power management specification by Texas Instruments. The NIC **42** is coupled through a physical layer **44** and a filter **46** to an RJ-45 jack **48**, and through a filter **50** to a AUI jack **52**.

Between the PCI Bus **10** and the ISA Bus **12**, an ISA/PCI backplane **54** is provided which include a number of PCI and ISA slots. This allows ISA cards or PCI cards to be installed into the system for added functionality.

Further coupled to the ISA Bus **12** is an enhanced sound system chip (ESS) **56**, which provides sound management through an audio in port **58** and an audio out port **60**. The ISA bus **12** also couples the ISA bridge **24** to a Super I/O chip **62**, which in the disclosed embodiment is a National Semiconductor Corporation PC87307VUL device. This Super I/O chip **62** provides a variety of input/output functionality, including a parallel port **64**, an infrared port **66**, a keyboard controller for a keyboard **68**, a mouse port for a mouse port **70**, additional series ports **72**, and a floppy disk drive controller for a floppy disk drive **74**. These devices are coupled through connectors to the Super I/O **62**.

The ISA bus **12** is also coupled through bus transceivers **76** to a flash ROM **78**, which can include both basic input/output system (BIOS) code for execution by the processor **32**, as well as an additional code for execution by microcontrollers in a ROM-sharing arrangement. The flash ROM **78** includes an ESCD sector for storing ESCD and associated variables. Details concerning ESCD may be

obtained from the Extended System Configuration Data Specification, Version 1.02, published on Feb. 14, 1994 by Compaq Computer Corporation, Intel Corporation, and Phoenix Technologies, Ltd.

The ISA bus **12** further couples the ISA bridge **24** to a security, power, ACPI, and miscellaneous application specific integrated circuit (ASIC) **80**, which provides a variety of miscellaneous functions for the system. The ASIC **80** includes security features, system power control, light emitting diode (LED) control, a PCI arbiter, remote wake up logic, system fan control, hood lock control, ACPI registers and support, system temperature control, and various glue logic. Finally, a video display **82** can be coupled to the AGP connector **18** for display of data by the computer system S. Again, a wide variety of systems could be used instead of the disclosed system S without detracting from the spirit of the invention.

Referring to FIG. **2**, a data flow diagram of an ESCD flash process in accordance with the present invention is shown. An operating system or application **100** may initiate an update of the ESCD sector **106** so as to store data to ESCD or associated variables. The operating system **100** for example may call ROM BIOS code to be executed by the processor **32** for performing an update of the ESCD sector **106** (FIGS. **2** & **3**). In accordance with the present invention, an update related to the ESCD sector **106** is achieved by copying data to an ESCD runtime buffer **112** during runtime or POST (Power-On-Self-Test). POST is designed to identify, test, and configure the computer system S in preparation for starting the operating system. During late POST, data associated with the ESCD sector **106** may be copied to an ESCD write buffer **110** (FIG. **3**). The set of operations collectively termed the POST operations are of the conventional type known in the art. During late POST, the contents of the ESCD write buffer **110** may be copied to the ESCD runtime buffer **112**. In the disclosed embodiment, data is copied a byte at a time. If the ESCD runtime buffer **112** has been updated, runtime code **116** (FIG. **3**) generates a software management interrupt (SMI) so as to pass the ROM image contained in the ESCD runtime buffer **112** to SMI code **104**. In the disclosed embodiment, the SMI code **104** is initially stored in the flash ROM **78** and copied to a volatile memory area during boot-up of the computer system S. The SMI code **104** flashes the ROM image to the ESCD sector **106** of the flash ROM **78**. The SMI code **104** includes an algorithm for flashing the ESCD sector **106** which is specific to the particular flash ROM **78**. An algorithm for flashing a flash ROM typically includes an erase sequence for erasing a flash ROM followed by a programming sequence for programming a flash ROM.

Referring to FIG. **3**, a data flow diagram of an ESCD flash process using an ESCD original buffer **108**, the ESCD write buffer **110**, and an ESCD runtime buffer **112** in accordance with the present invention is shown. In accordance with the present invention, the flash ROM **78** includes the ESCD sector **106**, POST code **114**, SMI code **104** for flashing a ROM image into the ESCD sector **106**, and code for executing the ESCD flash process. Referring to FIG. **4**, a flowchart of the ESCD flash process of FIG. **3** is shown. Following boot-up of the computer system S, POST is initiated. Beginning at step **200**, a ROM image of the ESCD sector **106** is copied to the ESCD original buffer **108** and the ESCD write buffer **110**. The ESCD original buffer **108** serves to maintain the original ROM image of the ESCD sector **106**. Next, in step **202**, the ESCD write buffer **110** may or may not be updated by POST. Step **202** is shown in broken line to reflect that the step is optional. If performed, step **202** is typically followed by a POST operation.

From either step **202** or **200**, control proceeds to step **204** of the POST operation wherein the contents of the ESCD write buffer **110** are copied to the ESCD runtime buffer **112** during POST **114**. Next, in step **206**, either the ESCD original buffer **108** or the ESCD sector **106** is compared to the ESCD runtime buffer **112**. The ESCD original buffer **108** contains a ROM image of the ESCD sector **106**. Either the ESCD original buffer **108**, ESCD write buffer **110**, or the ESCD sector **106** may be used to determine if the ESCD sector **106** has been updated. Since the ESCD original buffer **108** is a random access memory area, accessing the ESCD original buffer **108** is faster than accessing the ESCD sector **106**.

Control then proceeds to step **208** wherein it is determined if the ESCD runtime buffer **112** matches the compared buffer or sector. If the ESCD runtime buffer **112** does not match the compared buffer or sector, then control proceeds to step **210** wherein the SMI code **104** flashes the ROM image in the ESCD runtime buffer **112** into the ESCD sector **106**. A mismatch between the ESCD runtime buffer **112** and the compared buffer or sector indicates that the ESCD runtime buffer **112** has been updated. If the ESCD runtime buffer **112** matches the compared buffer or sector, control proceeds to step **212**. A match between the ESCD runtime buffer **112** and the compared buffer or sector indicates that the ESCD runtime buffer **112** has not been updated.

From step **210**, control proceeds to step **212**. Following step **208** or **210**, POST is exited and runtime begins. At step **212**, it is determined whether there is a write to ESCD data or associated variables. ESCD data and associated variables may change during runtime. If a write to ESCD data or associated variables is detected, control proceeds from step **212** to step **214**. In step **214**, the ESCD runtime buffer **112** is updated with the data written to ESCD or associated variables. The ESCD data or associated variables written may be new data or may be the same as previously stored data. Alternatively, it is contemplated that the ESCD runtime buffer **112** may be exclusively updated when a write of new ESCD data or associated variables is detected. From step **214**, control proceeds to step **216** wherein the SMI code **104** flashes the ROM image in the ESCD runtime buffer **112** into the ESCD sector **106**. This step is performed during a system management mode of the computer system S. From step **216**, control returns to step **212**. In this way, any write to ESCD or associated variables during runtime results in flashing a new ROM image into the ESCD sector **106**.

The use of the ESCD write buffer **110** and the ESCD runtime buffer **112** serves to isolate storage of ESCD and associated variables during POST from storage of ESCD and associated variables during runtime. The ESCD original buffer **108** and ESCD write buffer **110** both serve as ESCD POST buffers. The ESCD original buffer **108** maintains the original ROM image of the ESCD sector **106**, and the ESCD write buffer **110** reflects any updates to ESCD data and associated variables during POST. In the disclosed embodiment, the ESCD original buffer **108** and ESCD write buffer **110** are random access memory areas existing during POST. Further, in the disclosed embodiment, the ESCD runtime buffer **112** is a random access memory area existing during late POST and runtime. It should be understood that the ESCD buffers and the ESCD sector described store both ESCD data and associated variables.

In the disclosed embodiment, the associated variables include variables representing system features and ownership data. Specifically, the ownership data variables may include an ownership tag, a chassis serial number and manufacturing process numbers bytes. Use of an ownership

tag, chassis serial number, and manufacturing process number bytes are conventional and known in the art. The ownership tag may be an 80-byte standard ASCI character string protected by an administrator password. In the disclosed embodiment, the specific ownership tag used is, for example, "Property of_____," the particular vendor of the system S. An ownership tag is typically displayed during POST. Manufacturing process number bytes are used to track the last station of a manufacturing line which the computer system has been through.

In a conventional computer system, a segment selector from an operating system is used to locate ESCD. The segment selector provided by the operating system is configured for the particular operating mode—real mode, 16-bit protected mode, 32-bit protected mode—whichever is applicable. A segment selector in one operating mode is therefore different from a segment selector for another operating mode. Also, ESCD has typically resided in a portion of a ROM treated as an option ROM. In certain operating modes, however, availability of an option ROM is not guaranteed. For example, during runtime in a 32-bit protected mode, an operating system may not provide a segment selector to an option ROM, thereby preventing access to the option ROM during runtime. In accordance with the flash process of the present invention, a flash may be performed in any operating mode since there is no operating system intervention. Using SMI code in accordance with the present invention thus provides for flashing a non-volatile memory independent of the operating mode of the computer system.

Further, in a conventional computer system, updating ESCD has required accessing the non-volatile memory storing ESCD. The need to access a non-volatile memory each time ESCD is updated requires the transfer of ESCD across a relatively slow bus coupled to the non-volatile memory. In accordance with the present invention, a ROM image is flashed from the ESCD runtime buffer **112**. Accessing a random access memory area of the ESCD runtime buffer **112** as opposed to the ESCD sector **106** increases POST execution time since the ESCD runtime buffer **112** may be accessed without providing data across a relatively slow bus.

It should be understood that the flash process of the present invention may extend to areas of a non-volatile memory other than the ESCD sector. In accordance with the present invention, SMI code may be used to flash any area of the non-volatile memory. Further, it should be understood that a stack, pointers, and registers may be used in copying contents from one ESCD buffer to another ESCD buffer. It should also be understood that the flash process in accordance with the present invention encompasses flashing any non-volatile memory and therefore is not limited to flashing a read-only-memory.

The foregoing disclosure and description of the invention are illustrative and explanatory thereof, and various changes in the size, shape, materials, components, circuit elements, wiring connections and contacts, as well as in the details of the illustrated circuitry and construction and method of operation may be made without departing from the spirit of the invention.

What is claimed is:

**1**. A method of flashing an extended system configuration data (ESCD) storage area of a non-volatile memory of a computer system comprising the steps of:

copying a non-volatile memory image of an ESCD data storage area for storing ESCD and associated variables of a non-volatile memory to a first region of a power on self test (POST) buffer and a second region of the POST buffer;

copying contents of the second region of the POST buffer to an ESCD runtime buffer;

comparing the contents of the first region of the POST buffer or the ESCD data storage area of the non-volatile memory with the contents of the ESCD runtime buffer; and

flashing the non-volatile memory image in the ESCD runtime buffer into the ESCD data storage area during the system management mode of the computer system if the contents of the ESCD runtime buffer differ from the contents of the compared first region of the POST buffer or the ESCD data storage area.

**2**. The method of claim **1**, further comprising the steps of:

detecting an update to the ESCD runtime buffer during runtime; and

flashing the non-volatile image in the ESCD runtime buffer into the ESCD data storage area of the non-volatile memory.

**3**. The method of claim **1**, further comprising the step of:

updating the first region of the POST buffer during POST before the step of copying the contents of the second region of the POST buffer to the ESCD runtime buffer.

**4**. The method of claim **1**, wherein the non-volatile memory is a flash read-only memory.

**5**. The method of claim **1**, wherein the POST buffer is an area of random access memory existing during POST.

**6**. The method of claim **1**, wherein the ESCD runtime buffer is an area of random access memory existing during POST and runtime.

**7**. The method of claim **1**, wherein the associated variables comprise variables representing system features or ownership data.

**8**. The method of claim **1**, further comprising the step of:

entering a system management mode of the computer system from a 16-bit protected mode of the computer system before the step of flashing the non-volatile image.

**9**. The method of claim **1**, further comprising the step of:

entering a system management mode of the computer system from a 32-bit protected mode of the computer system before the step of flashing the non-volatile image.

**10**. The method of claim **1**, further comprising the step of:

entering a system management mode of the computer system from a real mode of the computer system before the step of flashing the non-volatile image.

**11**. A computer system for flashing an extended system configuration data (ESCD) storage area of a non-volatile memory comprising:

a non-volatile memory, comprising storage locations for:

an ESCD data storage area for storing ESCD data and associated variables;

a non-volatile memory BIOS storage area for storing BIOS code for generating an system management interrupt (SMI) flash for the ESCD data storage area of the non-volatile memory; and

an SMI storage area for storing SMI code for flashing the ESCD data storage area of the non-volatile memory responsive to generation of a system management interrupt by the BIOS code;

during Power on self test (POST) or during runtime responsive to a write to the ESCD data and associated variables and

a processor for executing the BIOS code.

**12**. The computer system of claim **11**, further comprising:

a volatile memory storing the BIOS code during execution of the BIOS code.

13. The computer system of claim 11, further comprising:

a volatile memory storing the SMI code during execution of the SMI code.

14. The computer system of claim 11, wherein the non-volatile memory is a flash read-only memory.

15. The computer system of claim 11, wherein the associated variables comprise variables representing system features or ownership data.

16. A computer system for flashing an extended system configuration data (ESCD) and associated variables into a non-volatile memory, comprising:

a non-volatile memory, comprising storage locations for:

an ESCD data storage area for storing ESCD data and associated variables;

power on self test (POST) code storage area for storing POST code for initializing the computer system;

an system management interrupt (SMI) code storage area for storing SMI code for flashing a non-volatile memory image into the ESCD data storage area; and

an ESCD flash code storage area for storing ESCD flash code for flashing said ESCD data and said associated variables during POST and runtime;

a volatile memory, comprising:

an ESCD original buffer for storing said ESCD data and said associated variables during POST;

an ESCD runtime buffer for storing said ESCD data and said associated variables during runtime;

an ESCD write buffer for providing said ESCD data and said associated variables to the ESCD runtime buffer during the POST; and

a processor executing the ESCD flash code to perform the steps of:

copying the non-volatile memory image of the ESCD data storage area to the ESCD original buffer and ESCD write buffer;

copying the contents of the ESCD write buffer to the ESCD runtime buffer;

comparing the contents of the ESCD original buffer or the ESCD data storage area with the contents of the ESCD runtime buffer; and

flashing the non-volatile memory image in the ESCD runtime buffer into the ESCD data storage area if the contents of the ESCD runtime buffer differ from the contents of the compared ESCD original buffer or the ESCD data storage area.

17. The computer system of claim 16, the processor executing the ESCD flash code to perform the further step of:

updating the ESCD write buffer with the POST code before the step of copying the contents of the ESCD write buffer to the ESCD runtime buffer.

18. The computer system of claim 16, the processor executing the ESCD flash code to perform the further steps of:

detecting an update to the ESCD runtime buffer; and

flashing the non-volatile memory image in the ESCD runtime buffer into the ESCD data storage area.

19. The computer system of claim 16, wherein the non-volatile memory is a flash read-only memory.

20. The computer system of claim 16, wherein the ESCD original buffer is a random access memory area existing during the POST.

21. The computer system of claim 16, wherein the ESCD write buffer is a random access memory area existing during the POST.

22. The computer system of claim 16, wherein the ESCD runtime buffer is a random access memory area existing during the POST and the runtime.

23. The computer system of claim 16, wherein the step of flashing the non-volatile memory image to the ESCD data storage area comprises the step of:

executing the SMI code for flashing a non-volatile memory image into the ESCD data storage area.

24. The computer system of claim 16, wherein the associated variables comprise variables representing system features or ownership data.

* * * * *